Managing Tomcat's Dependency on the Eclipse JDT Core Batch Compiler

Update: April 2019

For some time now, Eclipse has been publishing the JDT JAR to Maven central. Therefore the following simplified policy is now followed:

1. Use the latest available JAR from Maven central that supports the minimum Java version required by Tomcat.
2. POMs, build dependencies and any other dependency reference refers to the JAR available from Maven central.

Background

Tomcat depends on JDT to enable it to support the compilation of JSPs while running on a JRE. This ensures JSP compilation works out of the box. Tomcat can be configured to use javac if running on a JDK but that has proved problematic in the past - mainly around ensuring that tools.jar is on the class path.

For folks that use Tomcat with Maven in embedded mode it is important that the Maven dependencies are valid and correct. There have been problems in this area in the past such as bug 50604.

The root cause of the issue is that the small (~2MB) JDT JAR we use is not officially released to Maven Central by Eclipse. One or more volunteers upload the JAR and it typically appears 2-4 weeks after the Eclipse release.

There is a larger JAR (5.3MB) with contains the same functionality that is officially uploaded by Eclipse. It usually appears in Maven Central sooner that the smaller JAR.

Policy

1. Tomcat releases will always package the 'small' JDT JAR as they have done for several years.
2. The POMs used when uploading a release to Maven Central will always include references to valid JARs that are known to already exist in Maven central.
3. The POMs will continue to reference the 'small' Eclipse JAR as a non-optional dependency by default.
4. Tomcat's dependency on JDT will not normally be updated until the 'small' JAR is available in Maven Central. At that point all the dependencies (JAR that ships with Tomcat, JAR defined as dependency for Jasper JARs, JAR defined as dependency for emebedded JARs) will be updated together.
5. If we need to update the JDT dependency before the 'small' JAR is available in Maven central (e.g. security issue, access new features such as Java 9 support for testing etc.) then we will use the following fallback options for the POM references:
    - if the 'small' JDT JAR is not available, use the 'large' one;
    - if neither the 'small' nor the 'large' JDT JARs are available continue to reference the pre-update version of the 'small' JDT JAR.
6. The changelog should make clear which dependencies have been updated and which have not.